10-31-00

A

JC956 U.S. PTO 10/30/00

PTO/SB/05 (08-00)
Approved for use through 10/31/2002 OMB 0651-0032
U.S. Patent and Trademark Office, U.S. DEPARTMENT OF COMMERCE

Please type a plus sign (+) inside this box → [+]

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

JC825 U.S. PTO 09/699962 10/30/00

UTILITY PATENT APPLICATION TRANSMITTAL

(Only for new nonprovisional applications under 37 CFR 1.53(b))

Attorney Docket No.	MS150916.1
First Inventor	Matthew A. Goldberg
Title	STRING TEMPLATE PAGES FOR ....
Express Mail Label No.	EF185629361US

APPLICATION ELEMENTS

See MPEP chapter 600 concerning utility patent application contents.

1. [✔] Fee Transmittal Form (e.g., PTO/SB/17) *(Submit an original and a duplicate for fee processing)*
2. [ ] Applicant claims small entity status. See 37 CFR 1.27.
3. [✔] Specification [*Total Pages* 21] *(preferred arrangement set forth below)*
   - Descriptive title of the invention
   - Cross Reference to Related Applications
   - Statement Regarding Fed sponsored R & D
   - Reference to sequence listing, a table, or a computer program listing appendix
   - Background of the Invention
   - Brief Summary of the Invention
   - Brief Description of the Drawings *(if filed)*
   - Detailed Description
   - Claim(s)
   - Abstract of the Disclosure
4. [✔] Drawing(s) *(35 U.S.C. 113)* [*Total Sheets* 6]
5. Oath or Declaration [*Total Pages* 2]
   a. [✔] Newly executed (original or copy)
   b. [ ] Copy from a prior application (37 CFR 1.63 (d)) *(for continuation/divisional with Box 17 completed)*
      i. [ ] DELETION OF INVENTOR(S) Signed statement attached deleting inventor(s) named in the prior application, see 37 CFR 1.63(d)(2) and 1.33(b).
6. [ ] Application Data Sheet. See 37 CFR 1.76
7. [ ] CD-ROM or CD-R in duplicate, large table or Computer Program *(Appendix)*
8. Nucleotide and/or Amino Acid Sequence Submission *(if applicable, all necessary)*
   a. [ ] Computer Readable Form (CRF)
   b. Specification Sequence Listing on:
      i. [ ] CD-ROM or CD-R (2 copies); or
      ii. [ ] paper
   c. [ ] Statements verifying identity of above copies

ADDRESS TO: Assistant Commissioner for Patents
Box Patent Application
Washington, DC 20231

ACCOMPANYING APPLICATION PARTS

9. [✔] Assignment Papers (cover sheet & document(s))
10. [ ] 37 CFR 3.73(b) Statement *(when there is an assignee)* [ ] Power of Attorney
11. [ ] English Translation Document *(if applicable)*
12. [ ] Information Disclosure Statement (IDS)/PTO-1449 [ ] Copies of IDS Citations
13. [ ] Preliminary Amendment
14. [✔] Return Receipt Postcard (MPEP 503) *(Should be specifically itemized)*
15. [ ] Certified Copy of Priority Document(s) *(if foreign priority is claimed)*
16. [✔] Other: Express Mail Certificate

17. If a CONTINUING APPLICATION, *check appropriate box, and supply the requisite information below and in a preliminary amendment, or in an Application Data Sheet under 37 CFR 1.76:*

[ ] Continuation [ ] Divisional [ ] Continuation-in-part (CIP) of prior application No _____/_____

Prior application information *Examiner* _____ *Group / Art Unit.* _____

For CONTINUATION OR DIVISIONAL APPS only: The entire disclosure of the prior application, from which an oath or declaration is supplied under Box 5b, is considered a part of the disclosure of the accompanying continuation or divisional application and is hereby incorporated by reference. The incorporation can only be relied upon when a portion has been inadvertently omitted from the submitted application parts.

18. CORRESPONDENCE ADDRESS

[ ] *Customer Number or Bar Code Label* *(Insert Customer No. or Attach bar code label here)* or [✔] *Correspondence address below*

Name	Himanshu S. Amin				
Address	Amin, Eschweiler & Turocy, LLP				
	24th Floor, National City Center, 1900 East Ninth Street				
City	Cleveland	*State*	Ohio	*Zip Code*	44114
Country		*Telephone*	216-696-8730	*Fax*	216-696-8731

Name (Print/Type)	Himanshu S. Amin	*Registration No. (Attorney/Agent)*	40,894
Signature		*Date*	October 30, 2000

Burden Hour Statement. This form is estimated to take 0.2 hours to complete Time will vary depending upon the needs of the individual case Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, U.S Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS SEND TO Assistant Commissioner for Patents, Box Patent Application, Washington, DC 20231

PTO/SB/17 (09-00)
Approved for use through 10/31/2002 OMB 0651-0032
U S. Patent and Trademark Office, U.S DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number

FEE TRANSMITTAL for FY 2001

Patent fees are subject to annual revision.

TOTAL AMOUNT OF PAYMENT ($) 1,028.00

Complete if Known	
Application Number	
Filing Date	Herewith
First Named Inventor	Matthew A. Goldberg
Examiner Name	
Group Art Unit	
Attorney Docket No.	MS150916.1

METHOD OF PAYMENT

1. ☑ The Commissioner is hereby authorized to charge indicated fees and credit any overpayments to
   Deposit Account Number: 50-1063
   Deposit Account Name: Amin, Eschweiler & Turocy, LLP
   ☑ Charge Any Additional Fee Required Under 37 CFR 1 16 and 1 17
   ☐ Applicant claims small entity status. See 37 CFR 1.27
2. ☑ **Payment Enclosed:**
   ☑ Check ☐ Credit card ☐ Money Order ☐ Other

FEE CALCULATION

1. BASIC FILING FEE

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description	Fee Paid
101	710	201	355	Utility filing fee	710
106	320	206	160	Design filing fee	
107	490	207	245	Plant filing fee	
108	710	208	355	Reissue filing fee	
114	150	214	75	Provisional filing fee	

SUBTOTAL (1) ($) 710.00

2. EXTRA CLAIM FEES

		Extra Claims		Fee from below	Fee Paid
Total Claims	31	-20** = 11	X	18	= 198
Independent Claims	4	- 3** = 1	X	80	= 80
Multiple Dependent				0	= 0

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description
103	18	203	9	Claims in excess of 20
102	80	202	40	Independent claims in excess of 3
104	270	204	135	Multiple dependent claim, if not paid
109	80	209	40	** Reissue independent claims over original patent
110	18	210	9	** Reissue claims in excess of 20 and over original patent

SUBTOTAL (2) ($) 278.00

**or number previously paid, if greater; For Reissues, see above

FEE CALCULATION (continued)

3. ADDITIONAL FEES

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description	Fee Paid
105	130	205	65	Surcharge - late filing fee or oath	
127	50	227	25	Surcharge - late provisional filing fee or cover sheet	
139	130	139	130	Non-English specification	
147	2,520	147	2,520	For filing a request for *ex parte* reexamination	
112	920*	112	920*	Requesting publication of SIR prior to Examiner action	
113	1,840*	113	1,840*	Requesting publication of SIR after Examiner action	
115	110	215	55	Extension for reply within first month	
116	390	216	195	Extension for reply within second month	
117	890	217	445	Extension for reply within third month	
118	1,390	218	695	Extension for reply within fourth month	
128	1,890	228	945	Extension for reply within fifth month	
119	310	219	155	Notice of Appeal	
120	310	220	155	Filing a brief in support of an appeal	
121	270	221	135	Request for oral hearing	
138	1,510	138	1,510	Petition to institute a public use proceeding	
140	110	240	55	Petition to revive - unavoidable	
141	1,240	241	620	Petition to revive - unintentional	
142	1,240	242	620	Utility issue fee (or reissue)	
143	440	243	220	Design issue fee	
144	600	244	300	Plant issue fee	
122	130	122	130	Petitions to the Commissioner	
123	50	123	50	Petitions related to provisional applications	
126	240	126	240	Submission of Information Disclosure Stmt	
581	40	581	40	Recording each patent assignment per property (times number of properties)	40.00
146	710	246	355	Filing a submission after final rejection (37 CFR § 1.129(a))	
149	710	249	355	For each additional invention to be examined (37 CFR § 1 129(b))	
179	710	279	355	Request for Continued Examination (RCE)	
169	900	169	900	Request for expedited examination of a design application	
Other fee (specify) _____					

* Reduced by Basic Filing Fee Paid

SUBTOTAL (3) ($) 40.00

SUBMITTED BY

Name (Print/Type)	Himanshu S. Amin	Registration No. (Attorney/Agent)	40,984	Telephone	216-696-8730
Signature				Date	October 30, 2000

Complete (if applicable)

WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.

Burden Hour Statement: This form is estimated to take 0.2 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, U.S Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Assistant Commissioner for Patents, Washington, DC 20231

Atty. Docket No. MS150916.1

STRING TEMPLATE PAGES FOR GENERATING HTML DOCUMENT

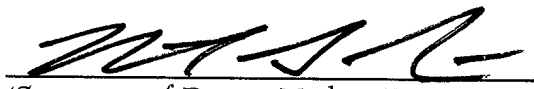
by

Matthew A. Goldberg

CERTIFICATION

I hereby certify that the attached patent application (along with any other paper referred to as being attached or enclosed) is being deposited with the United States Postal Service on this date October 30, 2000, in an envelope as "Express Mail Post Office to Addressee" Mailing Label Number EF185629361US addressed to the: Box Patent Application, Assistant Commissioner for Patents, Washington, D.C. 20231.

Himanshu S. Amin
(Typed or Printed Name of Person Mailing Paper)



(Signature of Person Mailing Paper)

Title: STRING TEMPLATE PAGES FOR GENERATING HTML DOCUMENT

Technical Field

The present invention relates to computer systems, and more particularly to a system and method for supply requested documents at a relatively quick rate.

Background of the Invention

Web documents are stored on web servers and are provided to client computers over the Internet upon receipt at the web server of a request for the document in the form of a uniform or universal resource locator ("URL"). The URL specifies the communications protocol by which the information is to be transferred and the Internet address of the host or web server upon which the document is stored. The URL may also specify a directory path and file name for the document. The communications protocol for the web is the hypertext transfer protocol ("HTTP"). Documents or pages stored on web servers and available over the web are generally formatted in a markup language. Markup language web documents contain text and a number of tags which provide instructions as to how the text should be displayed, which text should be hyperlinked to other documents, and where other types of content, including graphics and other images, video and audio segments, application programs or applets, image maps, and icons, should be retrieved from and displayed in the document. One of the most commonly used standardized markup languages is the Hypertext Markup Language ("HTML"), currently available in several versions. Other standardized markup languages include the Standard Generalized Markup Language ("SGML") and the Extensible Markup Language ("XML").

Conventionally, web servers receive requests from a client for a particular HTML document. The servers then load the HTML file (*e.g.,* an Application Service Provider (ASP) file) and parse the HTML file for script code. The script code is then interpreted and executed and the results sent back to the requestor, for example, by inserting the results into the loaded HTML file and transmitting the file and results back to the requestor. The problem is that the process of parsing, interpreting and executing the code is both time and resource intensive.

In order for a web server to serve many users at a reasonable rate, the web server must be able to supply the requested HTML pages very quickly and use as little system resources as possible. Web sites that dynamically generate web pages based on state and user input have the challenge of doing this in a scalable manner. The most efficient way to do so is through compiled code running in the same process as the web server. The most obvious place is for the HTML text to be part of the code itself, but this leads to disconnected, hard to read string constants that are difficult to maintain and that require a recompilation of the code whenever a change is made to the HTML text.

Accordingly, there is an unmet need in the art for a system and method that mitigates the above stated deficiencies with traditional HTML servers.

Summary of the Invention

A system and method is provided for supplying requested HTML pages at a relatively quick rate using limited resources. The present invention provides a solution to efficiently, clearly, and straightforwardly supplying text constants to code running in a server environment and using them to generate dynamically changing pages. The present invention employs an executable code component residing on a server system that inserts text templates including strings and arguments into requested HTML documents. The text strings general includes constant parts and variable parts. Variable parts change depending on state and user input. Each text template is considered a page and is loaded up at start up time by the executable code component, which parses and stores the constant parts of the text string into memory and retains the memory location of the text string. Additionally, the memory location of argument variables within the text constants can also be stored in memory. A request from a client causes the executable code component to load a copy of the appropriate HTML page, retrieve the appropriated text strings and arguments from the memory and insert the text strings and appropriate arguments into the loaded page. The copy is then transmitted back to the client. The executable code can also execute any code that was required by the HTML page to provide the desired results.

In one aspect of the invention, the constant part of the text strings residing in the templates can be modified without recompiling the code. Modifications to the templates can

be made while the executable code is still running. At the point when the executable code component loads a text template, it begins monitoring the text template for any changes made while running. Upon a change occurring, the text template is loaded and parsed again and the old information is replaced by the new information. Code referencing that page is automatically provided with the new information, thus, modifications occur dynamically.

To the accomplishment of the foregoing and related ends, the invention then, comprises the features hereinafter fully described and particularly pointed out in the claims. The following description and the annexed drawings set forth in detail certain illustrative embodiments of the invention. These embodiments are indicative, however, of but a few of the various ways in which the principles of the invention may be employed and the present invention is intended to include all such embodiments and their equivalents. Other objects, advantages and novel features of the invention will become apparent from the following detailed description of the invention when considered in conjunction with the drawings.

Brief Description of the Drawings

Fig. 1 illustrates a block diagram of a server system in accordance with one aspect of the present invention.

Fig. 2 illustrates a possible environment of the server system in accordance with one aspect of the invention.

Fig. 3 illustrates an example of a string template page in accordance with one aspect of the present invention.

Fig. 4 illustrates an example of employing intelligent IDs using data structures prior to referencing of the ID in accordance with one aspect of the present invention.

Fig. 5 illustrates an example of employing intelligent IDs using data structures after referencing of the ID in accordance with one aspect of the present invention.

Fig. 6 illustrates a flow diagram of one particular methodology for providing a server system in accordance with one aspect of the present invention.

Fig. 7 illustrates a flow diagram of one particular methodology for operation of an executable component in accordance with one aspect of the present invention.

Fig. 8 illustrates a block diagram of a computer system in accordance with an environment of the present invention.

Detailed Description of the Invention

The present invention is now described with reference to the drawings, wherein like reference numerals are used to refer to like elements throughout. The present invention is described with reference to a system and method for generating dynamically changing HTML documents at a relatively quick rate in response to a client request. The system and method provide separation of the constant text and the code embedded in the HTML documents. The present invention employs an executable component that includes all the basic functionality typically embedded in script code of a plurality of HTML documents. The dynamically generated text is broken down into string template pages. Each string template page can be authored as a single file and includes all the information necessary to complete a request. Editing the file can be done with the author's favorite document editing tool, and likewise viewing the file can be done in the typical manner as well, as long as meta-information in the page is included in a format that is considered "hidden" by the viewer.

A string template page includes one or more strings and each string includes a text constant and can include one or more argument variables. Each string template page is provided with identifiers for pages (*e.g.,* the file name), string or text constants (*e.g.,* text that does not change) and variable portions of strings (*e.g.,* arguments). The pages are read, parsed and processed into a data structure into memory by the executable component for efficient use at runtime. The executable component parses each page for where each string begins and ends and the strings unique identifier and where each argument should be inserted into the string constant and its argument number.

The present invention provides for separation of the text constants of HTML documents from the code embedded in the HTML documents. By separating the text constants out into individual files, including the minimal meta-information needed by the code to use them, readability, ease of authoring, and separation of text data and code are achieved. The needs of generating dynamic text can be served by breaking up the text into pages, strings, and places to insert arguments. To make the use of these templates at runtime

as fast as possible, the files are pre-processed at start-up time, efficiently storing information about where the pages and strings are located, with particular details about where they begin and end, and where to insert arguments. Intelligent IDs can take advantage of this efficiency, by allowing the code to reference objects by one ID only, defined at design time, while still gaining the direct access provided by pointers determined at run time.

Fig. 1 illustrates an example of a server system 10 for supplying requested HTML pages to a client system 20 according to the present invention. An executable component 14, for example, an executable binary component resides on the server 10. The executable component 14 is operable to load and parse a plurality of string template pages 12. The executable component parses string constants from the string template pages 12 by utilizing new string indicators provided in the pages and determines where variables are to be inserted by utilizing an insert argument indicator provided in the strings. The executable component 14 stores the string constants into a memory 18 and retains information relating to the memory location of each string constant and the insertion location of each variable. The executable component 14 also includes all the code necessary to respond to any request that may be made by the client 20. The executable component 14 also includes all the logic necessary regarding where and when strings should be used, whether or not and how often a string should be repeated.

The executable component 14 parses string constants from the string template pages 12 by utilizing new string indicators and determines where variables are to be inserted by utilizing insert argument indicators. The executable component 14 stores the string constants into a memory 18 and retains information relating to the memory location of each string constant and the insertion location of each variable. All the logic necessary to generate and insert the variable arguments into the string constants resides in the executable component. A variable can be a pointer to text, for example the user's name, or the current date, generated by the code. A variable can also be a pointer to another string in the page, for example, if there is an optional text constant which may or may not be used depending on a state. The argument can be authored as a string in the page and passed in as a variable only when applicable.

A monitoring system 16 is provided coupled to the executable component 14. The

monitoring system 16 monitors any changes to the string template pages 12. If any changes occur with any of the string template pages 12, the monitoring system 16 informs the executable component 14. The executable component 14 then reloads the changed string template page or pages, parses the string constants and variable locations, stores the new string constants in memory and retains the string constant and variable locations. The monitoring system 16 could be a configuration data source component, for example, in the Microsoft® Windows® Operating System environment, the configuration data source component type could be Windows Management Instrumentation (WMI), which is a support mechanism for management of systems in an enterprise. WMI allows developers to use a simple, consistent mechanism to query for information or configure settings on computers across an enterprise (*e.g.,* hardware settings, file change information, driver configuration, BIOS information, application settings, event log information). WMI allows for both hardware and software to be modeled. It is to be appreciated that other computer monitoring systems that provide configuration information may be employed to carry out the present invention.

The monitoring system 16 allows modification to the string template pages 12 while the executable component 14 is running. With the constant part of the text residing in the templates, completely separate from the code, changes can be made to the constant text without needing to recompile the code. After the code initially loads a string template page, the monitoring system 16 or the executable component 14 can begin monitoring that page or file for any changes made while running. When a change occurs, the page is loaded and parsed again, and the new information replaces the old. Code that references that page will now be given information about the new parse, such that modifications can be incorporated on the fly.

A request for a HTML document from a client 20 is received by the executable component 14. The executable component 14 executes any code relating to the request. The executable component 14 then loads up the string constants for a particular string template page 12 corresponding to the request using pointers pointing to the memory locations of the strings in the memory 18. The executable component 14 then retrieves the argument numbers, determines the appropriate variable based on the request or a state and inserts the

variable into the strings. The executable component 14 then returns the results of any executed code and the string constants with inserted variables therein. It is to be appreciated that the request that the executable component may insert the strings and arguments into a requested HTML page in place of the results that were conventionally provided by script. Alternatively, the executable component 14 *via* the code and the stored information of the string template pages 12 may serve up the entire HTML page and results requested without the need to load up any HTML page.

An example of a possible environment 30 for the system 10 is illustrated in Fig. 2. The environment 30 includes a local computer 32 having a display 34 and a user input device 36 through which an individual may interact with the local computer 32. The user input device 36, for example, may include a keyboard, a pointer device (*e.g.*, a mouse), a voice activated control device, a wireless input device, and/or other equipment or peripherals through which a user may interact with programs running on the local computer 32.

The local computer 32 is operatively coupled to a network 44, such as an Internet, an intranet, or another computer network. The local computer 32 is connected to the network 44, for example, over a telephone line 50 *via* a modem 38. Alternatively, the local computer 32 may be connected to the network 44 through another connection 42, such as an integrated services digital network (ISDN), T1, DS1 or other high speed telecommunications connections and an appropriate connection device, a television cable and modem, a satellite link, an optical fiber link, an Ethernet or other local area network technology wire and adapter card, radio or optical transmission devices, etc. It is to be appreciated that the invention also may be implemented in a browser environment for other public and private computer networks, such as a computer network of a commercial on-line service or an internal corporate local area network (LAN), an Intranet, or like computer network.

The local computer 32 runs software, including a browser 60, for unified browsing of electronic documents and other data from local sources as well as from the computer network 44. Specifically, documents for browsing with the browser software may reside as files of a file system stored in appropriate storage devices at the local computer 32 or reside at resources at a remote computer 68. The remote computer 68 is operatively coupled to the network 44 *via* connection 46, which may be in the form of a telecommunications connection

and appropriate device or any other communications link (including wired and wireless) to the network. By way of example, the remote computer 68 is an Internet-based server connected to the computer network 44 to provide one or more World Wide Web ("Web") sites to which the local computer 32 may connect.

By way of example, a plurality of HTML string template pages 52 resides at the remote computer 68 that conforms to HTML standards. It is to be appreciated that the browser software running at the local computer 32 may be capable of browsing documents having other data formats from the local computer or the remote computer 32. The HTML string template pages 52 include text constants that are logically grouped into pages and strings. A string is the largest text constant which will be used in total and with a pre-defined, constant number of variable parts, and a page is the collection of all the strings needed to complete one request. Editing the file can be done with a document editing tool, and likewise viewing the file can be done in the typical manner as well, as long as meta-information in the page is included in a format that is considered hidden by the viewer.

The browser software running locally on the local computer 32 displays a HTML document 62 in a window 64 or area of the local computer's display 34 allocated to the browser by the operating system. The window 64 includes a document display area 66 and user interface controls (not shown). The browser displays the HTML document 62 within the document display area 66 of the window 64, although other display types also may be used. The displayed document may include text and/or images generated by an executable component 54 in conjunction with the HTML string template pages 52.

In the example shown in Fig. 2, the display area 66 includes a display of a recipe portal web page. The recipe portal web page includes three selectable recipes labeled chicken pie, apple pie and pumpkin pie. A user may click on one of the recipes employing the user input device 36 to view a desired recipe. The selection of a desired recipe causes a request to be sent over the network 44 to the executable component 54 residing on the remote computer 68. The executable component 54 will execute any code necessary to responding to the request. The executable component 54 will then acquire the desired string constants from a memory 56 and insert the variable arguments (*e.g.,* the selected recipe) into the string constants. The executable component 54 can then transmit the desired HTML page and

appropriate code responses to the local computer 32 for display at the browser 60.

Fig. 3 illustrates an example of a string template page 70. The string template page 70 includes four strings and three variable arguments within the strings. A firsts string 72 includes a meta-tag labeled "BREAK" with a unique identifier name labeled "GUID1". The first string 72 is the header string. A second string 74 includes another meta-tag labeled "BREAK" with a unique identifier name labeled "GUID2". The second string 74 also includes two meta-tags labeled "INSERT" with corresponding argument numbers "ARG1" and "ARG2" within the HTML string. A third string 76 includes a meta-tag labeled "BREAK" with a unique identifier name labeled "GUID3". A fourth string 78 includes another meta-tag labeled "BREAK" with a unique identifier name labeled "GUID4". The fourth string 78 also includes a meta-tag labeled "INSERT" with an argument numbers "ARG3" within the HTML text of the string 78. The "INSERT" meta-tags define the beginning and end location of each string and the "INSERT" meta-tags define the location for inserting the different arguments by number. The executable component 54 can employ the "BREAK" and "INSERT" meta-tags to load the text constant information of the page 70 into memory 56 for use at runtime.

In one aspect of the invention, the executable component employs intelligent IDs. An intelligent ID is a globally scoped structure that contains two fields, the constant unique identifier and a pointer to the actual information (object). The executable code only needs to reference the intelligent ID when referring to an object. When the executable component starts up, every intelligent ID's pointer field is initially NULL. If an intelligent ID is referenced which still has a NULL pointer field, a search is made for the object referred to by the unique identifier field. When the object is found, the pointer to that object is stored in the intelligent ID. In all future references to the intelligent ID, the pointer field is found to be not NULL, and that pointer is used directly.

Figs. 4 and 5 illustrate an example of employing intelligent IDs in code of the executable component referencing the string template page 70 illustrated in Fig. 3. A data structure portion 80 includes a page data structure for the string template page 70 with a corresponding page ID. The page data structure includes a page name or ID and a pointer to the location of the page in memory. A plurality of string data structures are provided for the

string template page 70, which includes a first named "HEADER" and three remaining strings named "GUID2", "GUID3" and "GUID4". Each string data structure includes a pointer to the corresponding string in memory. Additionally, a list of argument number data structures are provided for the string template page 70. The argument numbers are labeled ARG1, ARG2 and ARG3. Each argument data structure includes a pointer to the corresponding argument in memory. In the example of Fig. 4, none of the IDs of the page, strings and arguments have been referenced and all of the pointers are set to "NULL". Fig. 5 illustrates the data structure portion 80 once the IDs have been referenced. After all of the IDs of the page, strings and arguments have been referenced, then all of the pointers have a pointer value for the data structure portion 80. In all future references to the IDs of the page strings and arguments the corresponding pointer can be used directly. It is to be appreciated that although in the present example, a single list of arguments for the string template page 70 is illustrated for simplicity purposes, a list of arguments may be provided for each string in the string template page 70.

In view of the structure described above with respect to Figs. 1-5, a methodology for providing the particular aspects of the invention may be better appreciated with respect to the flow diagrams of Figs. 6-7. While, for purposes of simplicity of explanation, the methodologies of Figs. 6-7 are shown and described as a series of steps, it is to be understood and appreciated that the present invention is not limited to the order of steps, as some steps may, in accordance with the present invention, occur in different orders and/or concurrently with other steps from that shown and described herein. Moreover, not all illustrated steps may be required to implement a methodology in accordance with an aspect the present invention.

Fig. 6 illustrates a methodology for providing a system for supply requested HTML pages in accordance with one aspect of the present invention. At step 90, any necessary request response code is defined. The string IDs and argument numbers to be used by the code and string template pages are defined at step 100. At step 110, the code is created using C, C++ or the like employing the necessary response code to any request and defining the data structures. Defining the data structures comprises defining the page IDs, string IDs for text constants and argument locations for the string variables. The executable code is then

compiled, by for example, a C, C++ or the like compiler at step 120 to provide a binary executable code. At step 130, the template string pages are created with page identifiers, string IDs for each string and argument numbers for variables to be inserted into the strings. At step 140, the compiled binary executable code is invoked.

Fig. 7 illustrates a methodology for the operation of the compiled binary executable code. At step 160, the compiled binary executable code loads the string template pages up into temporary memory or the like. The executable code then parses the string template pages and determines the beginning and end of each string and the argument location within each sting at step 170. At step 180, the executable code stores the string constants and argument numbers and retains the location of the string constants and argument numbers in memory, for example, by assigning pointers to data structures containing the string ID or argument number. The data structures include a string ID and pointer or an argument number and pointer. At step 190, the executable code determines if a request has been received from a client. If a request has been received from a client (YES), the executable component proceeds to step 195. At step 195, the executable component executes any code relating to the request, loads the strings (*e.g.,* into an HTML page), inserts the arguments into the strings and transmits the requested information back to the requestor. The executable component then returns to step 190. If a request has not been received from a client (NO) at step 190, the executable component determines if any changes have occurred in any of the string template pages at step 200. If a change has occurred in any of the string template pages (YES), the executable component reloads and parses the modified string template pages and retains the new string IDs and assigns new pointers to the data structure. The executable component then returns to step 190. If a change has not occurred in any of the string template pages (NO), The executable component returns to step 190.

With reference to Fig. 8, an exemplary system for implementing the invention includes a conventional personal or server computer 220, including a processing unit 221, a system memory 222, and a system bus 223 that couples various system components including the system memory to the processing unit 221. The processing unit may be any of various commercially available processors, including Intel x86, Pentium and compatible microprocessors from Intel and others, including Cyrix, AMD and Nexgen; Alpha from

Digital; MIPS from MIPS Technology, NEC, IDT, Siemens, and others; and the PowerPC from IBM and Motorola. Dual microprocessors and other multi-processor architectures also can be used as the processing unit 221.

The system bus may be any of several types of bus structure including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of conventional bus architectures such as PCI, VESA, Microchannel, ISA and EISA, to name a few. The system memory includes read only memory (ROM) 224 and random access memory (RAM) 225. A basic input/output system (BIOS), containing the basic routines that help to transfer information between elements within the computer 220, such as during start-up, is stored in ROM 224.

The computer 220 further includes a hard disk drive 227, a magnetic disk drive 228, *e.g.*, to read from or write to a removable disk 229, and an optical disk drive 230, *e.g.*, for reading a CD-ROM disk 231 or to read from or write to other optical media. The hard disk drive 227, magnetic disk drive 228, and optical disk drive 230 are connected to the system bus 223 by a hard disk drive interface 232, a magnetic disk drive interface 233, and an optical drive interface 234, respectively. The drives and their associated computer-readable media provide nonvolatile storage of data, data structures, computer-executable instructions, etc. for the server computer 220. Although the description of computer-readable media above refers to a hard disk, a removable magnetic disk and a CD, it should be appreciated by those skilled in the art that other types of media which are readable by a computer, such as magnetic cassettes, flash memory cards, digital video disks, Bernoulli cartridges, and the like, may also be used in the exemplary operating environment.

A number of program modules may be stored in the drives and RAM 225, including an operating system 235, one or more application programs 236, other program modules 237, and program data 238. The operating system 235 in the illustrated computer can be Microsoft Windows NT Server operating system, together with the before mentioned Microsoft Transaction Server, Microsoft Windows 95, Microsoft Windows 98 or Microsoft Windows 2000.

A user may enter commands and information into the computer 220 through a keyboard 240 and pointing device, such as a mouse 242. Other input devices (not shown)

may include a microphone, joystick, game pad, satellite dish, scanner, or the like. These and other input devices are often connected to the processing unit 221 through a serial port interface 246 that is coupled to the system bus, but may be connected by other interfaces, such as a parallel port, game port or a universal serial bus (USB). A monitor 247 or other type of display device is also connected to the system bus 223 via an interface, such as a video adapter 248. In addition to the monitor, computers typically include other peripheral output devices (not shown), such as speakers and printers.

The computer 220 may operate in a networked environment using logical connections to one or more remote computers, such as a remote server or client computer 249. The remote computer 249 may be a workstation, a server computer, a router, a peer device or other common network node, and typically includes many or all of the elements described relative to the computer 220, although only a memory storage device 250 has been illustrated in Fig. 8. The logical connections depicted in Fig. 8 include a local area network (LAN) 251 and a wide area network (WAN) 252. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets and the Internet.

When used in a LAN networking environment, the computer 220 is connected to the local network 251 through a network interface or adapter 253. When used in a WAN networking environment, the server computer 220 typically includes a modem 254, or is connected to a communications server on the LAN, or has other means for establishing communications over the wide area network 252, such as the Internet. The modem 254, which may be internal or external, is connected to the system bus 223 via the serial port interface 246. In a networked environment, program modules depicted relative to the computer 220, or portions thereof, may be stored in the remote memory storage device. It will be appreciated that the network connections shown are exemplary and other means of establishing a communications link between the computers may be used.

In accordance with practices of persons skilled in the art of computer programming, the present invention is described below with reference to acts and symbolic representations of operations that are performed by the computer 220, unless indicated otherwise. Such acts and operations are sometimes referred to as being computer-executed. It will be appreciated that the acts and symbolically represented operations include the manipulation by the

processing unit 221 of electrical signals representing data bits which causes a resulting transformation or reduction of the electrical signal representation, and the maintenance of data bits at memory locations in the memory system (including the system memory 222, hard drive 227, floppy disks 229, and CD-ROM 231) to thereby reconfigure or otherwise alter the computer system's operation, as well as other processing of signals. The memory locations where data bits are maintained are physical locations that have particular electrical, magnetic, or optical properties corresponding to the data bits.

The present invention has been illustrated with respect to a programming methodology and/or computer architecture and a particular example, however, it is to be appreciated that various programming methodology and/or computer architecture suitable for carrying out the present invention may be employed and are intended to fall within the scope of the hereto appended claims.

The invention has been described with reference to the preferred aspects of the invention. Obviously, modifications and alterations will occur to others upon reading and understanding the foregone detailed description. It is intended that the invention be construed as including all such modifications alterations, and equivalents thereof.

Claims

What is claimed is:

1. A system for servicing a client request comprising:

an executable component operable for receiving the request and executing code corresponding to the request, the executable component retrieving at least one text constant and inserting at least one variable argument result corresponding to the request into the at least one text constant and passing the at least one text constant and variable argument result to the client.

2. The system of claim 2, wherein the at least one text constant and the variable argument result reside in a memory location accessible by the executable component.

3. The system of claim 3, wherein the at least one text constant and the variable argument result are loaded into memory prior to runtime.

4. The system of claim 1, further comprising at least one string template page having at least one text constant.

5. The system of claim 4, the at least one string template page having at least one argument associated therewith.

6. The system of claim 4, the at least one text constant having at least one argument associated therewith.

7. The system of claim 6, the at least one text constant employing a new string indicator to identify each of the at least one text constant and a new argument indicator to identify each of the at least one argument.

8. The system of claim 4, the executable component loading the at least one string template page and parsing the at least one string template page to identify the at least one text constant, the executable component storing the at least one text constant into memory and retaining a unique identifier of the at least one text constant for retrieving the at least one text constant from memory at runtime.

9. The system of claim 8, the executable component also retaining a pointer to the memory location of the at least one text constant for retrieving the at least one text constant from memory at runtime.

10. The system of claim 8, the executable component parsing the at least one text constant to identify at least one argument associated with the at least one text constant, the executable component retaining a unique identifier number for the at least one argument for retrieving the at least one variable argument result at runtime.

11. The system of claim 10, the executable component also retaining a pointer to a location of the at least one argument within the at least one text constant for determining the location for inserting the at least one variable argument result into the at least one text constant at runtime.

12. The system of claim 8, further comprising a monitoring system for informing the executable component of any changes occurring in the at least one string template page, the executable component being adapted to reload, parse and store the changed at least one string template page upon being informed of any changes by the monitoring system.

13. The system of claim 12, the monitoring system being a configuration data source.

14. The system of claim 1, the executable component employing at least one

intelligent ID to retrieve the at least one text constant from memory.

15. The system of claim 1, the at least one text constant being an HTML text constant.

16. A method for providing a system for servicing a client request, comprising the steps of:

defining code for servicing the client request;

defining text constants to be used by the code;

assigning identifiers to the text constants;

providing at least one string template page with the defined text constants and assigned identifiers; and

providing executable code with the defined code and assigned identifiers of the text constants, the executable code being operable to load the defined text constants into a memory prior to runtime and reference the defined text constants employing the assigned identifiers to transmit the defined text constants at runtime to the client in response to a client request.

17. The method of claim 16, further comprising a step of defining argument variables to be inserted into the text constants and assigning identifier numbers to the argument variables.

18. The method of claim 17, the executable code being operable to reference the assigned identified numbers at runtime to provide at least one argument variable result and insert the argument variable result into the text constant at runtime.

19. A method for servicing a client request, comprising the steps of:

providing at least one string template page with at least one text constant and assigning an identifier corresponding to a respective text constant for each of the at least one

text constant;

parsing the at least one string template page for the at least text constant and retaining the corresponding assigned identifier;

storing the at least one text constant in a memory and retaining a memory location for the stored text constant;

receiving a request from a client;

retrieving the at least text string constant by utilizing the corresponding assigned identifier and the memory location for the stored text constant;

aggregating the retrieved text constant into a document; and

returning the document to the client.

20. The method of claim 19, further comprising a step of providing the at least one text constant of the at least one string template page with at least one argument and a corresponding argument number.

21. The method of claim 20, further comprising a step of parsing the at least one string template page for the location of the argument within the at least one text constant and the corresponding argument number and storing the location of the argument within the at least one text constant in the memory and retaining the corresponding argument number and memory location.

22. The method of claim 21, further comprising a step of retrieving the location of the argument within the at least one text constant from the memory location and utilizing the argument number and the location of the argument within the at least one text constant to insert an argument variable result into the retrieved text constant.

23. The method of claim 22, further comprising a step of providing each of the at least one text constants of the at least one string template page with a new string indicator to identify each of the at least one text constant and providing each of the arguments with a new

argument indicator to identify each of the arguments within the at least one text constant.

24. The method of claim 19, the step of retaining a memory location for the stored text constant comprising retaining a pointer to the memory location.

25. The method of claim 19, further comprising a step of monitoring any changes occurring in the at least one string template page and repeating the step of parsing and storing upon a detection of any change in the at least one string template page.

26. The method of claim 19, the at least one text constant being an HTML text constant.

27. A computer readable medium having computer-executable components comprising;

a first component operable for receiving a request from a client and executing code corresponding to the request, the first component retrieving at least one text constant residing in a memory corresponding to the request and passing the at least one text constant to the client.

28. The computer readable medium of claim 27, the first component being further operable to retrieve at least one variable argument result from the memory and insert the at least one variable argument result into the at least one text constant.

29. The computer readable medium of claim 27, further comprising at least one string template page, the first component being further operable to parse the at least one string template page for the at least one text constant and store the at least one text constant in the memory.

30. The computer readable medium of claim 27, further comprising a second

component adapted to monitor any changes in the at least one string template page and inform the first component of any changes in the at least one string template page.

31. The computer readable medium of claim 30, the first component being adapted to parse the changed at least one string template page for a changed at least one text constant upon being inform of any changes by the second component, the first component storing the changed at least one text constant in memory in place of the at least one text constant.

Abstract of the Invention

A system and method is provided for generating dynamically changing HTML documents at a relatively quick rate in response to a client request. The system and method provide separation of the constant text and the code embedded in the HTML documents. An executable component is employed that includes all the basic functionality typically embedded in script code of a plurality of HTML documents. The dynamically generated text is broken down into string template pages. Each string template page can be authored as a single file and includes all the information necessary to complete a request. A string template page includes one or more strings and each string is made up of a text constant and can include one or more argument variables. Each string template page is provided with identifiers for pages, text constants and variable portions of strings. The pages are read, parsed and processed into a data structure into memory by the executable component for efficient use at runtime.

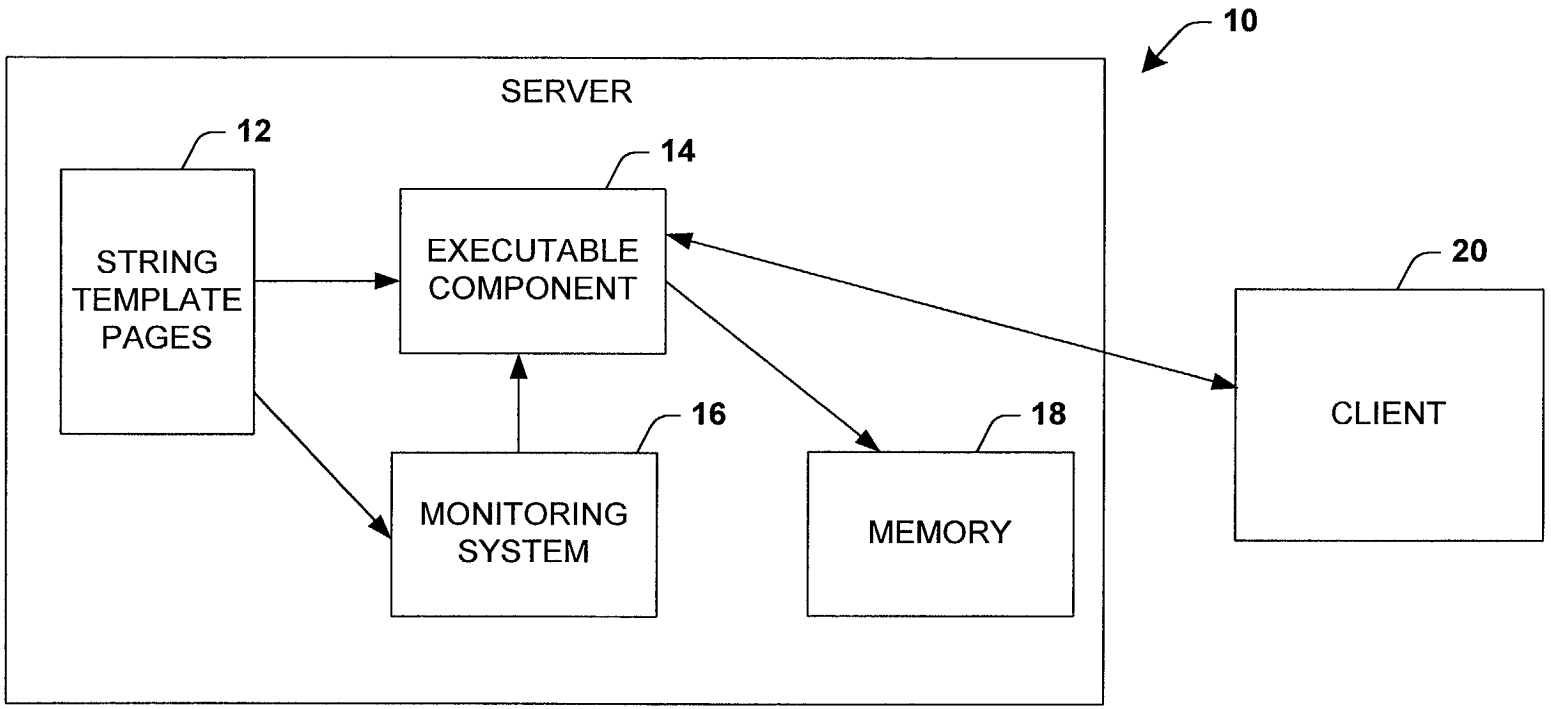


Fig. 1

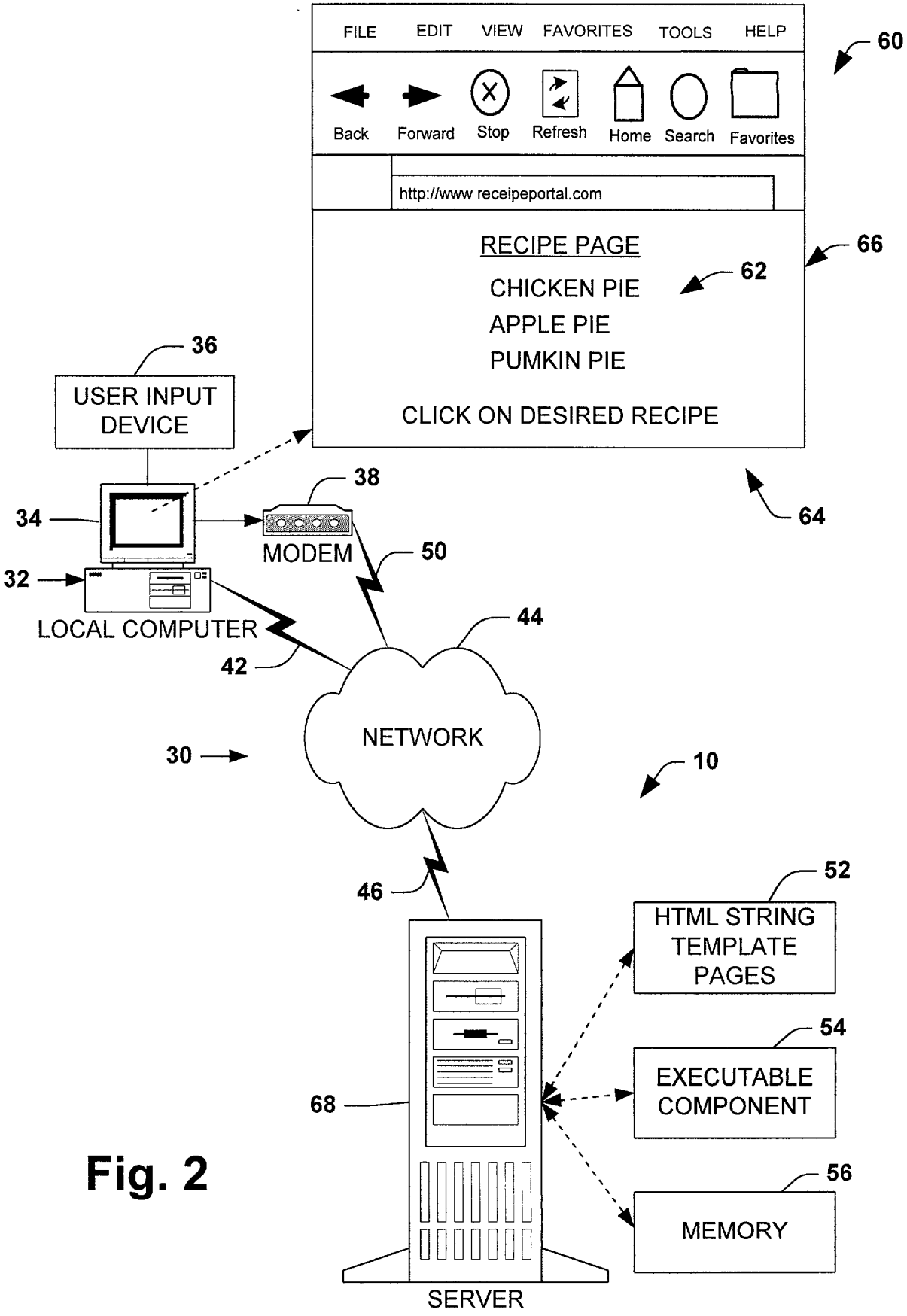

FILE
EDIT
VIEW
FAVORITES
TOOLS
HELP
Back
Forward
Stop
Refresh
Home
Search
Favorites
http://www receipeportal.com
RECIPE PAGE
CHICKEN PIE
APPLE PIE
PUMKIN PIE
CLICK ON DESIRED RECIPE
60
66
62
64
36
USER INPUT DEVICE
34
38
MODEM
50
32
LOCAL COMPUTER
44
42
NETWORK
30
10
46
52
HTML STRING TEMPLATE PAGES
54
EXECUTABLE COMPONENT
68
56
MEMORY
SERVER

Fig. 2

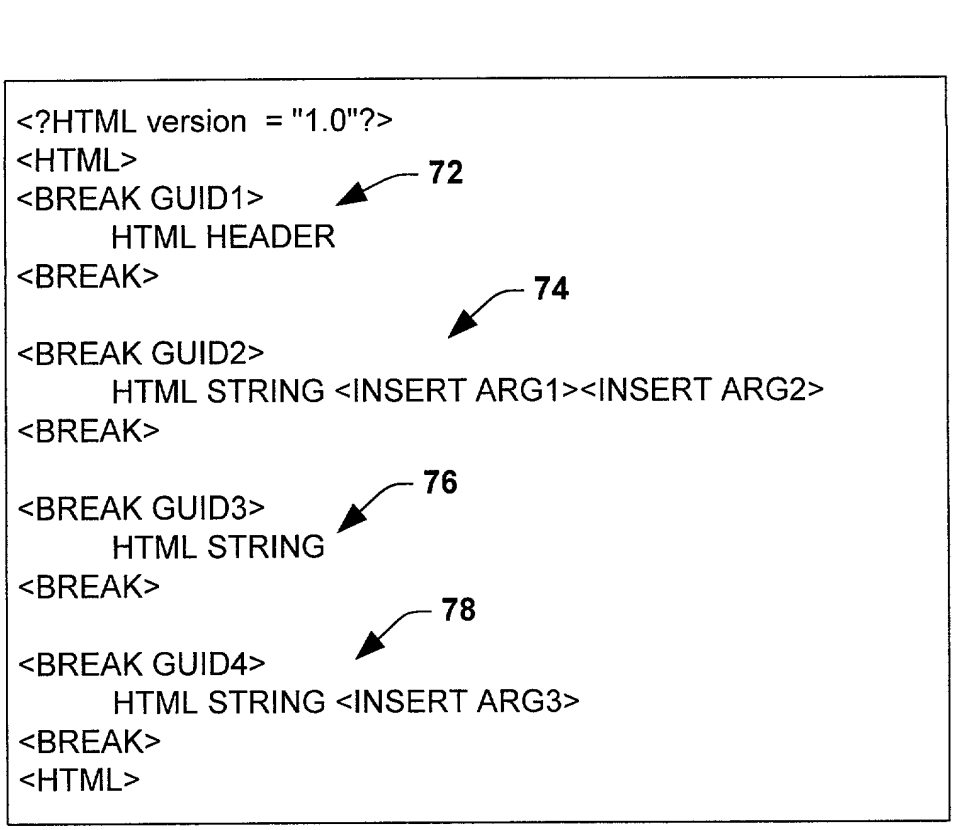


Fig. 3

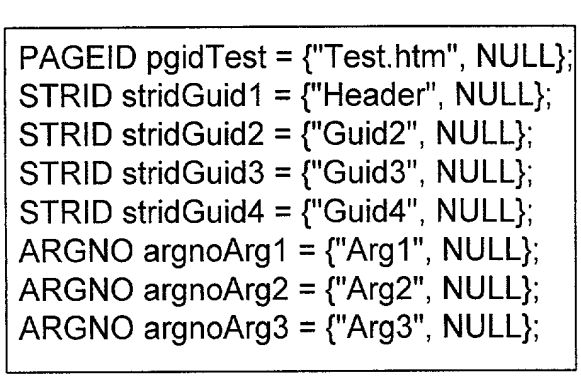


Fig. 4

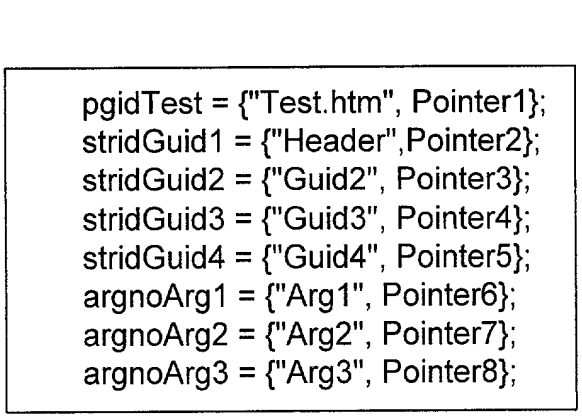


Fig. 5

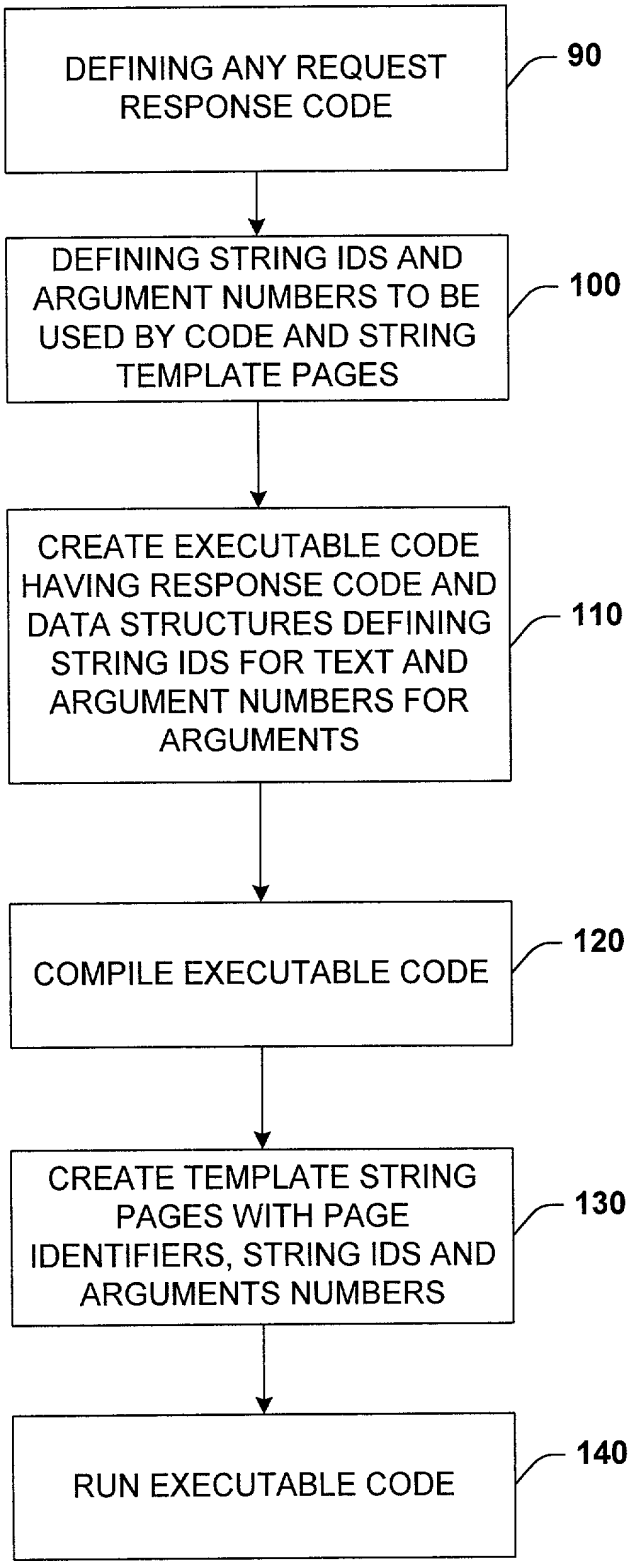

DEFINING ANY REQUEST RESPONSE CODE
90
DEFINING STRING IDS AND ARGUMENT NUMBERS TO BE USED BY CODE AND STRING TEMPLATE PAGES
100
CREATE EXECUTABLE CODE HAVING RESPONSE CODE AND DATA STRUCTURES DEFINING STRING IDS FOR TEXT AND ARGUMENT NUMBERS FOR ARGUMENTS
110
COMPILE EXECUTABLE CODE
120
CREATE TEMPLATE STRING PAGES WITH PAGE IDENTIFIERS, STRING IDS AND ARGUMENTS NUMBERS
130
RUN EXECUTABLE CODE
140

Fig. 6

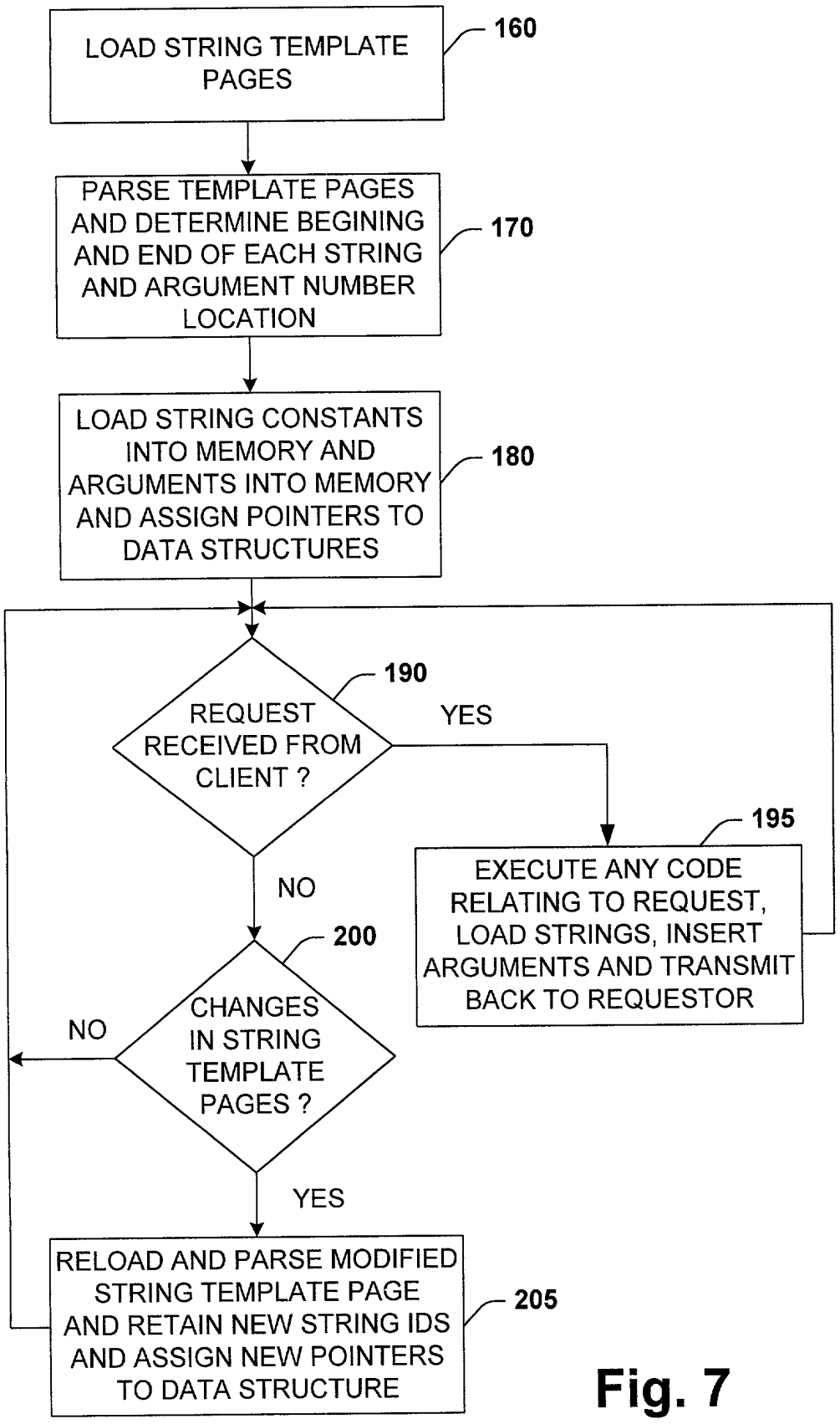

LOAD STRING TEMPLATE PAGES
160
PARSE TEMPLATE PAGES AND DETERMINE BEGINING AND END OF EACH STRING AND ARGUMENT NUMBER LOCATION
170
LOAD STRING CONSTANTS INTO MEMORY AND ARGUMENTS INTO MEMORY AND ASSIGN POINTERS TO DATA STRUCTURES
180
REQUEST RECEIVED FROM CLIENT ?
190
YES
NO
EXECUTE ANY CODE RELATING TO REQUEST, LOAD STRINGS, INSERT ARGUMENTS AND TRANSMIT BACK TO REQUESTOR
195
CHANGES IN STRING TEMPLATE PAGES ?
200
NO
YES
RELOAD AND PARSE MODIFIED STRING TEMPLATE PAGE AND RETAIN NEW STRING IDS AND ASSIGN NEW POINTERS TO DATA STRUCTURE
205

Fig. 7

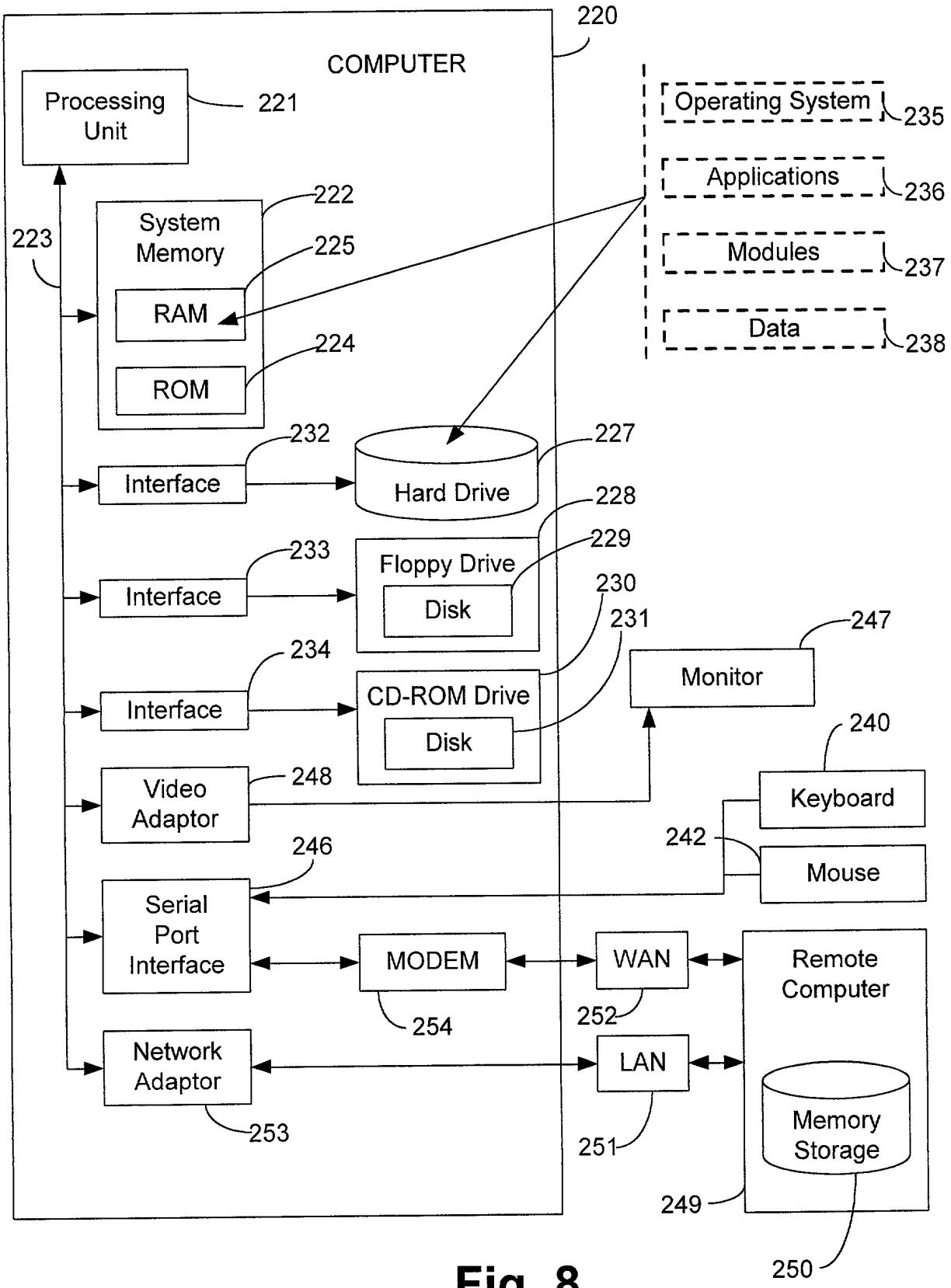

220
COMPUTER
Processing Unit
221
Operating System
235
Applications
236
Modules
237
Data
238
222
System Memory
223
225
RAM
224
ROM
232
Interface
227
Hard Drive
228
233
Interface
Floppy Drive
229
Disk
230
231
234
Interface
CD-ROM Drive
Disk
247
Monitor
240
Keyboard
242
Mouse
248
Video Adaptor
246
Serial Port Interface
MODEM
254
WAN
252
Remote Computer
Network Adaptor
253
LAN
251
Memory Storage
249
250

Fig. 8

COMBINED DECLARATION AND POWER OF ATTORNEY
(ORIGINAL, DESIGN, NATIONAL STAGE OF PCT)

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name, I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled: **STRING TEMPLATE PAGES FOR GENERATING HTML DOCUMENT**

the specification of which

(a) _X_ is attached hereto.

(b) ___ was filed on _____ as Serial No. 09 /_____ or Express Mail No._____, as Serial No. not yet known, and was amended on (if applicable).

(c) ___ was described and claimed in PCT International Application No. _____ filed on _____ and amended under PCT Article 19 on _____ (if any).

I hereby state that I have reviewed and understand the contents of the above identified specification, including the claims, as amended by any amendment referred to above.

I acknowledge the duty to disclose information which is material to patentability in accordance with Title 37, Code of Federal Regulations §1.56(a).

PRIORITY CLAIM

I hereby claim foreign priority benefits under Title 35, United States Code, §119 of any foreign application(s) for patent or inventor's certificate or of any PCT international application(s) designating at least one country other than the United States of America listed below and have also identified below any foreign application(s) for patent or inventor's certificate or any PCT international application(s) designating at least one country other than the United States of America filed by me on the same subject matter having a filing date before that of the application(s) of which priority is claimed.

(d) _X_ no such applications have been filed.

(e) ___ such applications have been filed as follows.

EARLIEST FOREIGN APPLICATION(S), IF ANY FILED WITHIN 12 MONTHS (6 MONTHS FOR DESIGN) PRIOR TO THIS U.S. APPLICATION

COUNTRY	APPLICATION NUMBER	DATE OF FILING (day, month, year)	PRIORITY CLAIMED UNDER 35, USC 119
_____	_____	_____	___ Yes ___ No
_____	_____	_____	___ Yes ___ No
_____	_____	_____	___ Yes ___ No

ALL FOREIGN APPLICATION(S), IF ANY FILED MORE THAN 12 MONTHS (6 MONTHS FOR DESIGN) PRIOR TO THIS U.S. APPLICATION

_____

_____

POWER OF ATTORNEY

As a named inventor, I hereby appoint the following attorney(s) and/or agent(s) to prosecute this application and transact all business in the Patent and Trademark Office connected therewith. (List name and registration number)

Himanshu S. Amin, Reg. No. 40,894; Gregory Turocy, Reg. No. 36, 952;
Christopher P. Harris, Reg. No. 43,660; Eric M. Highman,
Reg. No. 43,672; and Gary J. Pitzer, Reg. No. 39,334.

Katie E. Sako, Reg. No. 32,628 and Daniel D. Crouse, Reg. No. 32,022.

The undersigned to this declaration and power of attorney hereby authorizes the U.S. attorney(s) named herein to accept and follow instructions from:

Name(s) of authorized representative(s)_____

Address_____

as to any actions to be taken in the Patent and Trademark Office regarding this application without direct communication between the U.S. attorney(s) and the undersigned. In the event of a change in the person(s) from whom instructions may be taken, the U.S. attorney(s) will be so notified by the undersigned.

Send Correspondence To:

Himanshu S. Amin
AMIN, ESCHWEILER & TUROCY, LLP
24TH Floor, National City Center
1900 East 9TH Street
Cleveland, Ohio 44114

Direct Telephone Calls To:
(name and telephone number)

Himanshu S. Amin

(216) 696-8730

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with knowledge that willful false statements and the like are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the application or any patent issued therein.

Full name of **sole or first inventor**, if any: **Matthew A. Goldberg**
Inventor's signature: Matthew A Goldberg
Date: OCT. 27, 2000 Country of Citizenship: **U.S.**
Residence: **Bellevue, Washington**
Post Office Address: **14744 SE 63rd Place**
Bellevue, Washington 98006

CHECK FOR ANY OF THE FOLLOWING ADDED PAGE(S) WHICH FORM A PART OF THIS DECLARATION

X This declaration ends with this page.